- API –

The JDK Library is automatically read by the Java virtual machine when executing a Java program (both the String class and System class is within this library)

1. java.lang: Provides the basic functions of java.lang Java program even without explicitly specifying it.
2. java.util provides useful utility classes
3. java.io: Package providing the Input/Output functions
4. java.awt: Package that provides various components for graphical user interface (GUI)
5. java.awt.event: Package to control the events of awt components

API: Java application programming interfaces (APIs) are predefined software tools that easily enable interactivity between multiple applications.

- APIs in Java include classes, interfaces, and user Interfaces.

1. STRING

1.1. STRING ADDRESS

Comparing String value vs String address

```
(1)   String str1 = "Java";
(2)   String str2 = "Java";
(3)   String str3 = new String("Java");
```

- When a new String variable is crated without making a new object (1), any other variables that are created with the same value ((2) and (3)) all share the same value AND ADDRESS as "str1"

str1 == str2 == str3 (same address)
str1.equals(str2) (same value)
str1.equals(str3) (same value)

- Since str3 is a new object (3) it has a different address to str1 and str2

str1 ≠ str3 (different address)
str2 ≠ str3 (different address)
str1.equals(str3) (same value)
str2.equals(str3) (same value)

1.2. API METHODS

Type	Explanation	Example	Result

		`String str1 = "abcXabc";`
		`String str2 = new String("ABCXabc");`
		`String str3 = "  ja  va  ";`
		`String str4 = "안녕 Hello";`	

concat	Linking two strings	`System.out.println(str1.concat(str2));`	`abcXabcABCXabc`
Substring	Printing only a specific part of the string	`(1) System.out.println(str1.substring(3));` `(2) System.out.println(str1.substring(3, 5));`	`(1) Xabc` `(2) Xa`
length	Finding length of the string	`System.out.println(str1.length());`	`7`
upper /lower		`(1) System.out.println(str1.toUpperCase());` `(2) System.out.println(str1.toLowerCase());`	`(1) ABCXABC` `(2) abcxabc`
char_At	Counting from 0, find and print the x^{th} character	`System.out.println(str1.charAt(3));`	`X`

indexOf	Printing position of a particular character:	`(1) System.out.println(str1.indexOf('b');`
		`(2) System.out.println(str1.indexOf('b', 3));`
		`(3) System.out.println(str1.indexOf("abc"));`
		`(4) System.out.println(str1.indexOf("abc", 3));`
		`(5) System.out.println(str1.indexOf('z'));`
		`(6) System.out.println(str1.lastIndexOf('b'));`
			`(1) 1` (counts from the beginning)
			`(2) 5` (counts from the third character)
			`(3) 0`
			`(4) 4`
			`(5) -1` (if there is no such character in the String)

			(6) 5 (finds index of the last time 'b' appears)
Equals.	Returns true if equal	`System.out.println(str1.equals(str2));`	(1) false
	Ignores cases where str1 = str2	`System.out.println(str1.equalsIgnoreCase(str2));`	(2) true
		`Do {"x"` `} while ((!str1.equalsIgnoreCase(str2));`	Repeat "x" until str1 = str2
trim	Deletes space in front and back of the string	`System.out.println(str3.trim());`	ja va

replace	replace: Changing a char/string to a specific value	`System.out.println(str1.replace('a', '9'));`
		`System.out.println(str1.replace("abc", "#"));`
		`System.out.println(str3.replace(" ", ""));`
		`System.out.println(str1.replace("abc", "Z"));`
			(1) 9bcX9bc
			(2) #X#
			(3) Java (deleting all spaces)
			(4) ZXZ

	replace all: Changing all char's that fit a certain condition	`System.out.println(str4.replaceAll("[a-zA-Z]", ""));` `System.out.println(str4.replaceAll("[가-힣]", " "));  }`	(1) 안녕 (deleting all alphabets) (2) Hello (deleting all Korean letters)

1.3. HASHCODE

- A particular object has a particular hash code.
- If two objects are equal, their hash codes are the same. The reverse is not true.

```
String str1 = "Hello";
String str2 = "Hello";
∴ str1 and str2 have the same hash-code
```

- If the hash codes are different, then the objects are not equal for sure.

String Builder/buffer: When you want to add on to the existing String without making a new String (does not change the hash-code)

```
StringBuilder strBuilder = new StringBuilder("abc");
                System.out.println("strBuilder: " + strBuilder); //abc
                strBuilder.append("def");
                System.out.println("strBuilder: " + strBuilder); // abcdef
```

Both have same hash-code (does not change)

*String buffer works in the same way but takes a little more time than StringBuilder

1.4. STRING TOKENIZER

- Divides the string by space
-

```
StringTokenizer tokenizer1 = new StringTokenizer(A B C D E);
```

```
Tokenizer1.countTokens()
```

Counts the number of times this tokenizer's 'nextToken()' method can be called. In this case 5

```
while (tokenizer1.hasMoreTokens()) {
System.out.println(tokenizer1.nextToken());}
```

Tests if there are more tokens available from this token string

2. DATE

2.1. SIMPLE DATE FORMAT

```
(1) Date now1 = new Date();
(2) Calendar now2 = Calendar.getInstance();
```

```
    (3) GregorianCalendar now3 = new GregorianCalendar();
SETTING FORMAT:
  SimpleDateFormat sdf2 = new SimpleDateFormat("yyyy/MM/dd
                          (E) hh:mm:ss");

OUTPUT:
System.out.println(sdf.format(now1));
System.out.println(sdf.format(now2.getTime()));
System.out.println(sdf.format(now3.getTime()));
```

∴All give same output:

```
2022/04/18 (월) , 오후 05:14:42 (current date and time)
```

yyyy	four letter year
yy	two letter year
M	one letter month: 1- Jan, 2-Feb)
MM	two letter month: 01- Jan, 02-Feb
d	one letter day: 1, 2, 3
dd	two letter day: 01, 02, 03
E	day of the week: Mon, Tues, Wed
a	AM/PM
H	time- 24h format
h	time- 12h format
m	minute)
s	second
w	which weekth in the year?
D	which dayth in the year?

3. MATH

3.1. METHODS FROM CLASS MATHS

FUNCTION	COMMAND
Power	Math. pow(x, y) → x^y
Absolute value	Math. abs(x) → $\|x\|$
Square root	Math. sqrt(x) → $\sqrt{x}$
Minimum	Math. min(x, y) → returns larger value
Maximum	Math. max(x, y) → returns larger value

FUNCTION	COMMAND
random	Math. random () → Random floating number from 0~1
Absolute value	Math. random( ) ∗ 45 → Random integer from 0~45
Square root	(int) (Math. random( ) ∗ 45) → Random integer from 0~45
Minimum	Math. min(x, y) → returns larger value
Maximum	Math. max(x, y) → returns larger value

	1 decimal place	Units place	Tens place
Rounding UP (Math.ceil)	Math.ceil(9.15 * 10) / 10); ∴ 9.2	(int) Math.ceil(9.15) ∴ 10	(int) Math.ceil(85 / 10.0) * 10); ∴ 90
Rounding up/down (Math.round)	Math.round(9.15 * 10)/ 10.0) ∴ 9.2	Math.round(9.15) ∴ 9	Math.round(85 / 10.0) * 10) ∴ 90
Rounding DOWN (Math.floor)	Math.floor(9.15 * 10) ∴ 9.1	(int) Math.floor(9.15) ∴ 9	(int) Math.floor(85 / 10.0) ∴ 80

3.2. CLASS RANDOM

- The Random() method is a static method ad returns a random number of type double.
- Since it is static, it can be executed directly into the type Math.random() without creating an object.

An instance of this class is used to generate a stream of random numbers

```
Random random = new Random();
```

This constructor creates a new random number generator.

`random.nextInt()` → returns random integer $\pm 2^{32}$ with (approximately) equal probability.
`random.nextDouble()` → returns random double number $0 \sim 1$ with (approximately) equal probability.
`random.nextBoolean())` → returns random between true/false with (approximately) equal probability.
`random.nextInt(x)` → returns random integer between 0 (inclusive) and $x-1$ (exclusive) with (approximately) equal probability.
`random.nextInt(x) + 1)` → returns random between $\pm 2^{32}$ with (approximately) equal probability.

4. OBJECT

4.1. CLONING

- Method of duplicating the object itself to create a **new object**
- Only instances of classes that implement the Cloneable interface can be replicated. (`public class className implements cloneable`)
- The clone() defined in the Object class replicates only the **value** of the instance variable (cloned object has a different hashcode)

```
ClassName varName = (className) originalVarName.clone()
```

- In the object class, the method 'clone' needs to be overrien:

```
@Override
protected Object clone() throws CloneNotSupportedException {
return super.clone();
```

4.2. EQUALS

The equals method for class Object only returns true if both hashcode (var1==var2) and value (var1.equals(car2)) is equal. If you want to only compare one thing (address/value), you must override the 'Equals' method to compare the one thing u want. (see Source /ch14_api/src/com/lec/ex04_object/ex04_RectangleClass.java)

5. SCANNER

5.1. INTEGER

```
int intVarName = scanner.nextInt();
System.out.println("intVarName is " + intVarName);
```

5.2. STRING

```
String stringVarName = scanner.next();
System.out.println("stringVarName is " + stringVarName);
```

5.3. VALUES WITH SPACES (address, full name…)

The nextLine() method of the java.util.Scanner class scans from the current position until it finds a line separator delimiter. The method returns the String from the current position to the end of the line.

```
System.out.print("variable where user inputs spaces?");
scanner.nextLine();(1)
String spaceVarName = scanner.nextLine();(2)
System.out.println("spaceVarName is " + spaceVarName);
```

At step (1), it deletes all the \n in the buffer and at step (1) it only inputs the address

6. OTHERS

6.1. WRAPPER

Wrapper class: Transforms basic data into object data (each new object data has a different address regardless of their value)

TYPE	BASIC DATA TYPE (1)	OBJECT DATA TYPE (2)	CONSTRUCTING NEW OBJECT	USING NEW OBJECT
Integer	int	Int	(1): `int i = 10` (2): `Integer iObj = new Integer(10);`	(1) `i` (2) `Obji.intValue()` or `Obji`
Decimal	double	Double	(1): `double i = 10.1;` (2): `Double iObj = new Double(10.1);`	(1) `i` (2) `Obji`

Converting data types:

- String → Integer: `int i = Integer.parseInt("10");`
- Integer → String: `String monthStr = String.valueOf(12);`

6.2. TIMER

- The Timer object causes the TimerTask object to become operational at a certain time, or to operate the TimerTask object every certain amount of time.
- Since the TimerTask class is an abstract class, a class that inherits from the TimerTask class must be created

Example:

```
System.out.println("Start");
        Timer timer = new Timer(true);
        TimerTask task1 = new TimerTaskEx1();
        TimerTask task2 = new TimerTaskEx2();
        timer.schedule(task1, 2000);     // run task1.run() after 2 seconds (2000 millisecs = 2 seconds)
        timer.schedule(task2, 1000, 500); // After 1 second, run task2.run() every 0.5 seconds
        Thread.sleep(5000);              // After 5 seconds, stop
        System.out.println("END");
```

6.3. DECIMAL

num = 12300000000000L = 1.23 * (10^{13})		
`DecimalFormat df0 = new DecimalFormat ("00000000")`		`System.out.println(df0.format(num))` ∴12300000000000
`DecimalFormat df1 = new DecimalFormat ("########")`		`System.out.println(df1.format(num))` ∴12300000000000
`DecimalFormat df2 = new DecimalFormat("0,000.000")`	Every 3 digits add comma, and up to 3.d.p	`System.out.println(df2.format(num));` ∴12,300,000,000,000.000
`DecimalFormat df3 = new DecimalFormat("#,###.##%");`	Since ##% has three digits, print up to 3.d.p	`System.out.println(df3.format(num));` ∴1,230,000,000,000,000%
`DecimalFormat df4 = new DecimalFormat("#.##E00");`		`System.out.println(df4.format(num));` ∴1.23E13